AMATH 482 Homework 1

Gonzalo Ferrandez Quinto

January 24, 2020

1 Introduction and Overview

On this writing, we are trying to solve for the location of a submarine which emits acoustic noise while its navigating. Sound amplitude data is provided at discrete points of a 3D grid of the ocean where its believed the submarine can be found. Unfortunately, the submarine is moving at the same time the data collected contains background white noise which is hiding its signature. To overcome this, measurements are taken every 30 minutes over a 24 hour window. Therefore, my objective is to filter out the background noise using all this measurements in time and uncover the position of the submarine.

2 Theoretical Background

2.1 Fourier Series

To accomplish my objective, I will be using the Fast Fourier Transform and a filtering function applied to the frequency space to get rid of the background noise. To build up to theoretical reasons that allow this, first we need to understand what the Fourier series is. Fourier introduced in 1822 the concept of representing a by a trigonometric series of sines and cosines of different oscillatory frequency k:

$$f(x) = \frac{a_0}{2} + \sum_{k=1}^{\infty} \left(a_n \cos \frac{kx\pi}{L} + b_n \sin \frac{kx\pi}{L} \right) \quad x \in (-L, L]. \tag{1}$$

Therefore by finding the corresponding values of a_0, a_k and b_k; it is possible to fully characterize a function. This can be accomplished by exploiting the orthogonality properties of sines and cosines over the $(-L, L]$ interval. It should be noted that the cosines and sines need to be 2π periodic in the interval. That is the reason that the frequencies are being scaled by $2\pi/2L$. This subtlety will be passed on in the algorithmic implementation.

2.2 Fourier Transform

This brings us to the idea of a Fourier Transform. Which given a function $f(x)$ with $x \in R$, it is possible to define its Fourier transform $f\hat{(}k)$ such as:

$$\begin{aligned} f\hat{(}k) &= \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{\infty} f(x) e^{-ikx} dx \\ f(x) &= \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{\infty} f\hat{(}k) e^{ikx} dk \end{aligned} \tag{2}$$

Therefore by taking the inner product of a function $f(x)$ with the negative complex exponential over all space, one is able to create another function that depends on k, $f\hat{(}k)$. Where a given k produces an output proportional to the amplitude of the sinusoidal function of that frequency that can be used to characterize $f(x)$. In this case, the output will be a complex number in which the modulus is proportional to the amplitude of the function, analogous to the coefficients a_k and b_k. The phase of this complex number will give

us how much does this corresponds to $cos(kx)$ or $sin(kx)$. In addition, it is possible to recover the original function by applying inverse Fourier transform.

2.3 Discrete Fourier Transfrom

For our given set up, it is impossible to apply the Fourier Transform as we are only given a discrete number of points. Therefore the Discrete Fourier Transform (DFT) needs to be used. Given by:

$$\hat{x}_k = \frac{1}{N} \sum_{k=0}^{N-1} x_n e^{\frac{kn2\pi}{N}} \tag{3}$$

Where x_n is a vector of length N. In addition, due to the discretization only a finite number of frequencies are present. With $\hat{x}_k = [-N/2, -N/2+1, ..., -1, 0, 1, ..., N/2+1]$ and also being a vector of length N. Also due to periodicity, the frequencies $-k$ and $k+N$ are the same, this effect is called aliasing. In this case, for my implementation I will not using the DFT algorithm but the Fast Fourier Transform (FFT) which takes considerably less computational power with $\mathcal{O}(Nlog(n))$ complexity instead of $\mathcal{O}(N^2)$. In addition, both of this algorithms posses an inverse that returns functions from the frequency domain.

2.4 Translational shift

In addition, a property of the Fourier transform that I will rely heavily on is the translational shift. As for some function $f(x)$ with x_0 being a real value, $\mathcal{F}[f(x-x_0)] = \mathcal{F}[f(x)]e^{-ikx_0} = \hat{f(k)}e^{-ikx_0}$. Where $\mathcal{F}$ represents applying the Fourier transform. Moreover, this property has significant physical consequences, as although the transform gets multiplied by a complex number, its magnitude is still the one of $\hat{f(k)}$ for a given k. Therefore in the frequency domain, the absolute values of shifted functions will look the same.

2.5 White Noise Filtering

2.5.1 filter function

Another property of the Fourier Transform, its the ability to filter out noise in the frequency domain. If the frequency at which the desired signal peaks is known, it is possible to filter the data by multiplying it with a function around this point. In this case, the function will decay the contributions of frequencies at which we know the signal is not travelling. Therefore after doing the inverse Fourier transform(IFT), the peak of the signal will be revealed in spatial domain. A commonly used function is a Gaussian such as:

$$G(k) = e^{-\tau(k-k_0)^2} \tag{4}$$

Were τ is inversely proportional to the width of the function and k_0 determines where the function is centered at. For the submarine problem, the analogous 3D Gaussian function will be need to be used of the form:

$$G(kx, ky, kz) = e^{-\tau((kx-kx_0)^2+(ky-ky_0)^2+(kz-kz_0)^2)} \tag{5}$$

In addition, artificially trying to create a peak in the frequency domain using the white noise and the Gaussian function, will result in no defined peak in the spatial domain. As the IFT will be able to identify is mainly composed of noise contributions.

2.5.2 finding peak frequency

In order to filter out part of the noise, finding the peak frequency at which the submarine's acoustic noise is propagating is previously needed. Although this peak is being masked by the white noise background, there is a way around this.

White noise is described as adding a normally distributed random variable with zero mean to the frequency domain. If data is available over multiple realizations(measurements at different times) in the frequency domain, part of this noise will average closer to zero. Revealing better details about the denoised wave if we average the different realizations, especially at what frequencies does it peak. It is important to underline that this is possible due to the translational shift property of the Fourier Transform. As the submarine could be moving around, therefore the sound wave produced by it will be located at different points in space. The problem is that averaging will delete all spacial shift dependence. This is an example of a manifestation of the Heisenberg Uncertainty Principle, where knowing the specific frequency of the signal will create spacial uncertainty.

3 Algorithm Implementation and Development

To achieve the objective of locating the submarine in space the following steps are *followed*:

1. Determine the peak frequencies at which the sound produced by the submarine travels
2. Filter the data in the frequency domain around the the peak frequencies
3. Find the maximum sound amplitude locations, where the submarine will supposedly closely located

For the first objective, the input data comes in a 2D array and needs to be reshaped into a 64x64x64 3D matrix. Inside a for loop, this is done using the function `reshape()` and each one of the time realizations goes through `fftn()` (N-dimensional fast Fourier). As I explained in part 2.3, this is similar to the DFT, but instead it is a 3D dimensional version of the FFT that needs a 3D input. Therefore it will characterize the whole space in frequency space for each one of the realization. In addition, on each iteration of the for loop, the realizations are summed up. This will cancel out the noise as it averages closer to zero, as I previously explained in part 2.4. After that I average the matrix over the number of realizations and use `max()` to find the value and index of the 3D array with highest amplitude corresponding to the peak amplitude. Then I use `ind2sub` to find the matrix subscripts that correspond to this peak frequency. In addition each one of these subscripts will correspond to the index in the vector k that includes all the frequencies. With the `[I1,I2,I3]` values corresponding to the x,y and z directions stored.

Some subtleties for this section is that when the space is divided in discrete points, usually one uses powers of 2. This is what works best with the FFT algortihm, that is the reason we use n = 64, which is 2^6. In addition, we need to re scale the frequency values by $\frac{2\pi}{2L}$ as the FFT assumes 2π periodicity in the interval from $[-L, L]$. In addition, our space domain in this case is a 3D cube of side length 20 going from -10 to 10. In addition, in many cases when displaying the FFT, one nneds to appy the `fftshift()` fuction. As matlab displays the frequencies in ascending order starting with the positive and then continuing with the negative ones.

For part 2, the peak frequency is now known, therefore I am able to apply the 3D Gaussian filter. In this case this is created as specified in 2.5.1 using an external fuction called `createfilter()` that I created 2. It basically fills a 3D array with the corresponding amplitude of the 3D Gaussian function for which each cell would correspond to the the data after going only through the `fftn()` fuction. In this case I am using a $\tau = 0.1$. It should be noted that changing this value will alter the output location data in a small fraction.

After this, each one of the realizations goes through a for loop that FFT transforms the data into frequency space. Afterwards the filter is applied by multiplying the transformed array with the filter array. As explained in 2.5.1, this will diminish the contribution of the noise. After this process, the realization goes through the `ifftn()` function which inverses the Fourier transform. Now due to the filtering, a maximum amplitude of sound and its index can be found using the `max()` function as before with the frequencies. These index input goes through the `ind2sub` function to output the index location in each one of the spatial direction vectors, `[x,y,z]`. This values are saved over each realization in the `locationXYZ` array, where each row corresponds to a spatial direction.

Finnaly I use the`plot3()` function to create a 3d plot of the movement of the submarine. In addition, the

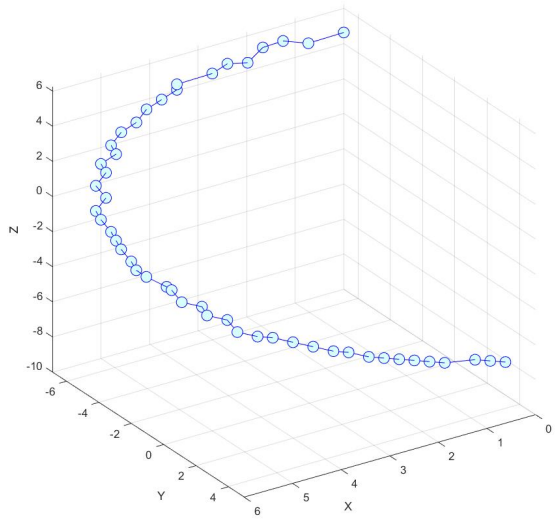

Figure 1: The submarine is going upward in the Z direction

table is created and saved using the `table()` function.

4 Computational Results

In the first part of the code, I was able to compute the peak frequencies for the sound propagation of the submarine. this corresponded to values of $[-6.115, 5.3407, 2.1991]$ for each one of the x,y and z directions respectively.

In the second part of the code I was able thanks to the filtering of the data to find the location of submarine over time. This is shown in the graph 1. In addition it should be noted that the movement of the submarine is in the positive Z direction.

Finally, I also included a table1 with the X-Y coordinates of the submarine over time. If the submarine needs to be neutralized, a P-8 Poseidon sub-tracking aircraft should be sended to the (0.625,-5) location. As said before, this values are slightly dependant of the value of τ for the filter, therefore they are just approximations.

5 Summary and Conclusions

During this whole project, the value of Fourier Transform has been valued in applications. Many of this analysis could have not been done, if it was not for some of the properties that this possesses such as time shift Independence. In addition, I also heavily relied on the fact that the noise was white noise, if not other methods would have been needed to be used.

Appendix A MATLAB Functions

Add your important MATLAB functions here with a brief implementation explanation. This is how to make an **unordered** list:

- `y = linspace(x1,x2,n)` returns a row vector of `n` evenly spaced points between `x1` and `x2`.

Table 1: This table contains the X-Y coordinates of submarine at the specified times

X	**Y**	**Time(hours)**
0	3.125	0
0.3125	3.125	0.5
0.625	3.125	1
1.25	3.125	1.5
1.5625	3.125	2
1.875	3.125	2.5
2.1875	3.125	3
2.5	3.125	3.5
2.8125	3.125	4
3.125	2.8125	4.5
3.4375	2.8125	5
3.75	2.5	5.5
4.0625	2.1875	6
4.375	1.875	6.5
4.6875	1.875	7
5	1.5625	7.5
5	0.9375	8
5.3125	0.625	8.5
5.3125	0.3125	9
5.625	0	9.5
5.625	-0.625	10
5.625	-0.9375	10.5
5.9375	-1.25	11
5.9375	-1.875	11.5
5.9375	-2.1875	12
5.9375	-2.8125	12.5
5.9375	-3.125	13
5.9375	-3.4375	13.5
5.9375	-4.0625	14
5.9375	-4.375	14.5
5.625	-4.6875	15
5.625	-5.3125	15.5
5.3125	-5.625	16
5.3125	-5.9375	16.5
5	-5.9375	17
5	-6.25	17.5
4.6875	-6.5625	18
4.375	-6.5625	18.5
4.0625	-6.875	19
3.75	-6.875	19.5
3.4375	-6.875	20
3.4375	-6.875	20.5
2.8125	-6.5625	21
2.5	-6.5625	21.5
2.1875	-6.25	22
1.875	-6.25	22.5
1.5625	-5.9375	23
1.25	-5.3125	23.5
0.625	-5	24

- `fftn()` is the N-dimensional analogous function of the `fft()`
- `ifftn()` is the IFFT for N-dimensions
- `max()` returns the maximum value of an array
- `ind2sub()` gives the subscript values given the index for an array
- `reshape()` reshapes a given matrix into another set of specified dimensions

Appendix B MATLAB Code

```
% Clean workspace
 clear all; close all; clc
 load('subdata.mat') %load data matrix
 L = 10; % spatial domain
 n = 64; % Fourier modes
 x2 = linspace(-L,L,n+1); x = x2(1:n); y =x; z = x; %create 3D axis arrays with 64 points
 k = (2*pi/(2*L))*[0:(n/2 - 1) -n/2:-1]; %create frequency array and re scale them to be 2pi periodic

 %use a for loop and a matrix to sum all the realizations
 SumUnt = zeros(n,n,n);
 for t = 1:49
     Un(:,:,:) = reshape(subdata(:,t),n,n,n); %reshape input data into 3D matrix form
     Unt = fftn(Un); %Using Fast Fourier transform in 3D
     SumUnt = SumUnt + Unt;
 end
 SumUnt = SumUnt/49; %normalizing the sumation of the realizations in frequency space
 [C,I] = max(SumUnt(:)); %finding peak frequencies in 3D
 [I1,I2,I3] = ind2sub(size(SumUnt),I); %match the indexes to subscripts in a 3D matrix

 % Find peak frequencies
 MaxKx = k(I1);
 MaxKy = k(I2);
 MaxKz = k(I3);

%creates Gaussian Filter fuction
tau = 0.1 %inversely proportional to width of filter
GaussianFilter = createfilter(k,k,k,MaxKx,MaxKy,MaxKz,tau,n);

locationXYZ = zeros(3,49)
for t=1:49
Un(:,:,:) = reshape(subdata(:,t),n,n,n);
%use Gaussian filter on the frequency domain of the data
Unt = fftn(Un);
Uf = ifftn(Unt.*GaussianFilter);
[C,I] = max(Uf(:)); %find peak of amplitude in filtered sound
[I1,I2,I3] = ind2sub(size(Uf),I); %match index to 3D matrix
locationXYZ(1,t) = x(I1);
locationXYZ(2,t) = y(I2);
locationXYZ(3,t) = z(I3);
end
%Plots the location of the submarine over time
plot3(locationXYZ(1,:),locationXYZ(2,:),
locationXYZ(3,:),'-o','Color','b','MarkerSize',10,
'MarkerFaceColor','#D9FFFF')
grid on
xlabel('X')
ylabel('Y')
zlabel('Z')

%creates and saves the table with the locations of the submarine
time = 0:0.5:24
T = table(locationXYZ(1,:)',locationXYZ(2,:)',time')
writetable(T,'table.csv','Delimiter',' ')
type 'table.csv'
```

Listing 1: Main code from HWsubmarine.m

```
%
%function that creates the a 3D Gaussain fuction, given the frequency
%discretization in 3 coordinates and its corresponding peaks. Also needs a
%tau
function result = createfilter(kx,ky,kz,MaxKx,MaxKy,MaxKz,tau,n)
filter = rand(n,n,n);
f = @(Kx,Ky,Kz)(exp(-tau*((Kx-MaxKx).^2+(Ky-MaxKy).^2+(Kz-MaxKz).^2)));
for i=1:n
for j =1:n
for l=1:n
filter(i,j,l) = f(kx(i),ky(j),kz(l));
end
end
end
result = filter;
end
```

Listing 2: fuction that creates filter for HWsubmarine.m